ASSIGNMENT 1

COMP202, Winter 2020

Due: Sunday, Feb. 9th, 11:59pm

Please read the entire PDF before starting. You must do this assignment individually.

Question 1:	60 points
Question 2:	40 points
	100 points total

It is very important that you follow the directions as closely as possible. The directions, while perhaps tedious, are designed to make it as easy as possible for the TAs to mark the assignments by letting them run your assignment, in some cases through automated tests. While these tests will never be used to determine your entire grade, they speed up the process significantly, which allows the TAs to provide better feedback and not waste time on administrative details. Plus, if the TA is in a good mood while he or she is grading, then that increases the chance of them giving out partial marks. :)

Up to 30% can be removed for bad indentation of your code as well as omitting comments, or poor coding structure.

To get full marks, you must:

- Follow all directions below
  - In particular, make sure that all file names, function names, and variable names are **spelled exactly** as described in this document. Otherwise, a 50% penalty will be applied.
- Make sure that your code runs.
  - Code with errors will receive a very low mark.
- Write your name and student ID as a comment in all .py files you hand in
- Name your variables appropriately
  - The purpose of each variable should be obvious from the name
- Comment your work
  - A comment every line is not needed, but there should be enough comments to fully understand your program

Part 1 (0 points): Warm-up

Do **NOT** *submit this part, as it will not be graded. However, doing these exercises might help you to do the second part of the assignment, which will be graded. If you have difficulties with the questions of Part 1, then we suggest that you consult the TAs during their office hours; they can help you and work with you through the warm-up questions. You are responsible for knowing all of the material in these questions.*

Warm-up Question 1 (0 points)
Practice with Number Bases:
We usually use base 10 in our daily lives, because we have ten fingers. When operating in base 10, numbers have a `ones` column, a `tens` column, a `hundreds` column, etc. These are all the powers of 10.

There is nothing special about 10 though. This can in fact be done with any number. In base 2, we have each column representing (from right to left) 1,2,4,8,16, etc. In base 3, it would be 1,3,9,27, etc.

Answer the following short questions about number representation and counting.

1. In base 10, what is the largest digit that you can put in each column? What about base 2? Base 3? Base n?
2. Represent the number thirteen in base 5.
3. Represent the number thirteen in base 2.
4. What is the number 11010010 in base 10?

Warm-up Question 2 (0 points)
Logic:

1. What does the following logical expression evaluate to?

   ```
   (False or False) and (True and (not False))
   ```

2. Let a and b be boolean variables. Is it possible to set values for a and b to have the following expression evaluate as *False*?

   ```
   b or (((not a) or (not a)) or (a or (not b)))
   ```

Warm-up Question 3 (0 points)
Expressions: Write a program `even_and_positive.py` that takes an integer as input from the user and displays on your screen whether it is true or false that such integer is even, positive, or both.

An example of what you could see in the shell when you run the program is:

```
>>> %Run even_and_positive.py
Please enter a number: -2
-2 is an even number: True
-2 is a positive number: False
-2 is a positive even number: False

>>> %Run even_and_positive.py
Please enter a number: 7
7 is an even number: False
7 is a positive number: True
7 is a positive even number: False
```

Warm-up Question 4 (0 points)

Conditional statements: Write a program `hello_bye.py` that takes an integer as input from the user. If the integer is equal to 1, then the program displays `Hello everyone!`, otherwise it displays `Bye bye!`.

An example of what you could see in the shell when you run the program is:

```
>>> %Run hello_bye.py
Choose a number: 0
Bye bye!

>>> %Run hello_bye.py
Choose a number: 1
Hello everyone!

>>> %Run hello_bye.py
Choose a number: 329
Bye bye!
```

Warm-up Question 5 (0 points)

Void Functions: Create a file called `greetings.py`, and in this file, define a function called `hello`. This function should take one input argument and display a string obtained by concatenating *Hello* with the input received. For instance, if you call `hello("world!")` in your program you should see the following displayed on your screen:
`Hello world!`

- Think about three different ways of writing this function.
- What is the return value of the function?

Warm-up Question 6 (0 points)

Void Functions: Create a file called `drawing_numbers.py`. In this file create a function called `display_two`. The function should not take any input argument and should display the following pattern:

```
  22
2   2
   2
 2
22222
```

Use strings composed out of the space character and the character '2'.

- Think about two different ways of writing this function.
- Try to write a similar function `display_twenty` which displays the pattern '20'

Warm-up Question 7 (0 points)

Fruitful Functions: Write a function that takes three integers `x`, `y`, and `z` as input. This function returns True if `z` is equal to 3 or if `z` is equal to the sum of `x` and `y`, and False otherwise.

Warm-up Question 8 (0 points)

Fruitful Functions: Write a function that takes two integers `x`, `y` and a string `op`. This function returns the sum of `x` and `y` if `op` is equal to `+`, the product of `x` and `y` if `op` is equal to `*`, and zero in all other cases.

Part 2

The questions in this part of the assignment will be graded.
The main learning objectives for this assignment are:

- Correctly create and use variables.
- Learn how to build expressions containing different type of operators.
- Get familiar with string concatenation.
- Correctly use `print` to display information.
- Understand the difference between inputs to a function and inputs to a program (received from the user through the function `input`).
- Correctly use and manipulate inputs received by the program from the function `input`.
- Correctly use simple conditional statements.
- Correctly define and use simple functions.
- Solidify your understanding of how executing instructions from the shell differs from running a program.

Note that the assignment is designed for you to be practicing what you have learned in class up to Tuesday Jan 28th. For this reason, **you are NOT allowed** to use what we have not seen in class (for example, you cannot use loops!). You will be heavily penalized if you do so.

For full marks on both questions, make sure to add the appropriate documentation string (docstring) to *all* the functions you write. The docstring must contain the following:

- The type contract of the function.
- A description of what the function is expected to do.
- At least 3 examples of calls to the function (beside the `display_welcome_menu` functions which always display the same content). You are allowed to use *at most* one example per function from this pdf.

Please note that all the examples are given as if you were to call the functions from the shell.

Question 1: Vending Machine (60 points)

For this part of the assignment you will write a program that simulate a virtual vending machine. Your program must retrieve inputs from a user interacting with the vending machine. The purpose of the program is to compute the amount of change to be returned by the machine given the amount of money inserted by the user and the cost of the item they have chosen. Assume that the change will be composed of toonies ($2), loonies ($1), quarters ($0.25), dimes ($0.10), and nickels ($0.05).

Note that, to be considered correct, your program must provide the user with the most convenient exact change (i.e. the exact amount with the fewest coins).

To achieve this, create a file called `vending_machine.py`. At the beginning of the file create five global variables indicating the amount of coins available in the vending machine. The machine has a total of $25 distributed as follows:

- $10 in toonies
- $5 in loonies
- $5 in quarters
- $3 in dimes
- $2 in nickels

The five global variables must be called with the following names: `TOONIES`, `LOONIES`, `QUARTERS`, `DIMES`, and `NICKELS`.

For full marks, all the following functions must be part of your program:

- `display_welcome_menu`: This function takes no input and *displays* a welcoming message as well as a list with all the options available to the customer. That is,

  ```
  >>> display_welcome_menu()
  Welcome to the COMP 202 virtual Vending Machine.
  Here are your options:
  1. Candy bar $2.95
  2. Cookies $3.90
  3. Soda $4.00
  4. Chips $3.90
  5. No snacks for me today!
  ```

  You can personalize the welcome message, but the options available must be exactly the same as listed above.

- `get_snack_price`: This function takes an integer and input representing the choice of the customer after seeing the menu. The function *returns* the price (in cents) associated to the choice made. If the input is not a number between 1 and 4 (both included) then the function returns 0. For example:

  ```
  >>> get_snack_price(2)
  390
  >>> get_snack_price(21)
  0
  ```

- `get_num_of_coins`: This function takes three positive integers as input, the first indicating the amount of money needed (in cents), the second indicating the value of a coin (in cents), and the third indicating the number of coins of the given value available in the machine. The function *returns* the maximum number of coins of the given value that can be used to work toward achieving the target amount. For example:

  ```
  >>> get_num_of_coins(1000, 100, 8)
  8
  >>> get_num_of_coins(550, 200, 4)
  2
  >>> get_num_of_coins(234, 30, 20)
  7
  ```

- `compute_and_display_change`: This function takes as input one non-negative integer representing the change (in cents) that the vending machine should give back to the customer. Using the coins available in the machine, the method computes the most convenient exact change. If the machine has enough coins to make the change, then the function *displays* the corresponding information on the screen and *returns* `True`. Otherwise, the function must not display anything and *returns* `False`. For example:

  ```
  >>> b = compute_and_display_change(185)
  Here is your change:
   toonies x 0
   loonies x 1
   quarters x 3
   dimes x 1
   nickels x 0
  >>> b
  True
  ```

```
>>> b = compute_and_display_change(40)
Here is your change:
 toonies x 0
 loonies x 0
 quarters x 1
 dimes x 1
 nickels x 1
>>> b
True

>>> b = compute_and_display_change(590)
Here is your change:
 toonies x 2
 loonies x 1
 quarters x 3
 dimes x 1
 nickels x 1
>>> b
True

>>> b = compute_and_display_change(2600)
>>> b
False

>>> b = compute_and_display_change(123)
>>> b
False
```

- `operate_machine`: This function takes no inputs and returns no value. The function performs the following tasks in this following order:
  - It displays the menu of the vending machine
  - It retrieves an integer from the user indicating their choice
  - If no snack item was chosen, then it displays a goodbye message and terminate its execution. Otherwise, it displays the price (in cents) of the chosen item and continues to the next step.
  - It retrieves the cash (in dollars) provided by the user.
  - It rounds the amount provided by the user to two decimals and displays it (in cents).
  - If the amount provided if not a multiple of 5 then it displays a message indicating that no pennies are accepted and terminates its execution.
  - If the amount provided is not enough to cover the price of the snack, then it displays a messages indicating that the cash is not enough and terminates its execution.
  - It displays the change (in cents) that the user should expect from the machine
  - It computes and displays the change to be provided. If it was not possible to provide the change needed, it displays a message indicating that the machine does not have enough coins. Otherwise, it displays a message indicating the order was successfully processed.

  As long as you maintain the correct meaning, you are free to personalize the messages displayed by your vending machine. Below you can find a couple of examples of interactions with the vending machine. Note that the inputs provided from the user appear in a different font and in blue.

```
>>> operate_machine()
Welcome to the COMP 202 virtual Vending Machine.
Here are your options:
1. Candy bar $2.95
2. Cookies $3.90
3. Soda $4.00
4. Chips $3.90
5. No snacks for me today!

Please select your choice: 5

Nothing for you today. Thank you for stopping by!

>>> operate_machine()
Welcome to the COMP 202 virtual Vending Machine.
Here are your options:
1. Candy bar $2.95
2. Cookies $3.90
3. Soda $4.00
4. Chips $3.90
5. No snacks for me today!

Please select your choice: 2

The item of your choice costs 390 cents


Enter your money: $3.45555

You inserted 346 cents

We do not accept pennies. Come by another time!

>>> operate_machine()
Welcome to the COMP 202 virtual Vending Machine.
Here are your options:
1. Candy bar $2.95
2. Cookies $3.90
3. Soda $4.00
4. Chips $3.90
5. No snacks for me today!

Please select your choice: 2

The item of your choice costs 390 cents


Enter your money: $3.4999999

You inserted 350 cents

You do not have enough money. Come by another time!
```

```
>>> operate_machine()
Welcome to the COMP 202 virtual Vending Machine.
Here are your options:
1. Candy bar $2.95
2. Cookies $3.90
3. Soda $4.00
4. Chips $3.90
5. No snacks for me today!

Please select your choice: 2

The item of your choice costs 390 cents


Enter your money: $7.55

You inserted 755 cents

You should receive back 365 cents
Here is your change:
 toonies x 1
 loonies x 1
 quarters x 2
 dimes x 1
 nickels x 1

It was a pleasure doing business with you!

>>> operate_machine()
Welcome to the COMP 202 virtual Vending Machine.
Here are your options:
1. Candy bar $2.95
2. Cookies $3.90
3. Soda $4.00
4. Chips $3.90
5. No snacks for me today!

Please select your choice: 2

The item of your choice costs 390 cents


Enter your money: $29.80

You inserted 2980 cents

You should receive back 2590 cents

The machine does not have enough coins for your change. Come by another time!
```

For full credit you should never be repeating code, but rather calling helper functions! You are welcome to add additional functions if you think this can increase the readability of your code.

Question 2: Pizza Calculator (40 points)

Last week, Johnny went to a pizza shop and asked for a 12-inch pizza. The shop owner said they didn't have any 12-inch pizzas left and suggested selling Johnny two 6-inch pizzas for the same price. Johnny accepted the offer.

After Johnny got home, his girlfriend told him that he had been ripped off. The area of a circle is

$$Area = \pi r^2$$

Where r is the radius of the circle and π is roughly 3.14. So, the 12-inch pizza, with a radius of 6 inches, has an area of

$$3.14 * 6^2 = 113.04$$

whereas a 6-inch pizza, with a radius of 3 inches, has an area of

$$3.14 * 3^2 = 28.26$$

This means that two 6-inch pizzas only have an area of $28.26 * 2 = 56.52$! Thus, two 6-inch pizzas are equivalent to only half of a 12-inch pizza!

Feeling slightly traumatized by this experience, Johnny decides to hire you to write a Python program to prevent being ripped off again by unethical pizza shops.

For this question, create a program called `pizza_calculator.py` and write the following functions:

- `display_welcome_menu`: This function takes no input and *displays* a welcoming message as well as a list with all the options available to Johnny (the user).

  That is,

  ```
  >>> display_welcome_menu()
  Welcome to the COMP 202 fair pizza calculator!
  Please chose one of the following modes:
  A. "Quantity mode"
  B. "Price mode"
  ```

- `get_fair_quantity`: This function takes two positive integers as input representing the diameters of two pizzas. The function *returns* an integer indicating the minimum number of smaller pizzas Johnny must order in order for him to get *at least* the same amount of pizza as one large pizza.

  For example:

  ```
  >>> get_fair_quantity(14, 8)
  4
  >>> get_fair_quantity(8, 14)
  4
  >>> get_fair_quantity(14, 5)
  8
  >>> get_fair_quantity(5, 5)
  1
  ```

- `get_fair_price`: This function takes as input an integer indicating the diameter of the large pizza, a real number indicating the price of the large pizza, an integer indicating the diameter of the small pizza, and one last integer indicating the number of small pizzas to be ordered. You can assume the inputs will always be positive numbers in the order just specified. The function *returns* the total price Johnny should be paying to buy the smaller pizzas such that the amount of pizza per dollar is the same as that of the larger pizza. The return value should be a real number with no more than 2 digits after the decimal point.

For example:

```
>>> get_fair_price(12, 10.0, 6, 2)
5.0
>>> get_fair_price(14, 9.55, 8, 4)
12.47
```

- `run_pizza_calculator`: This function takes no inputs and returns no value. The function performs the following tasks in this following order:
  - It displays the menu of the pizza calculator
  - It retrieves a string indicating the choice of the user.
  - If "Quantity mode" was selected, then it retrieves two more integers indicating the diameter of the large and the small pizza. It then display the quantity of small pizzas the user should buy to be satisfied.
  - If "Price mode" was selected, then it retrieves the following:
    1. The size (diameter) of the larger pizza (integer)
    2. The price of the larger pizza (real number)
    3. The size (diameter) of the smaller pizza (integer)
    4. The number of smaller pizzas (integer)

    It then displays the fair price the user should pay for the specified number of small pizzas.
  - If any other mode is selected, the function should display a message indicating that such mode is not supported.

  You can find below a couple of example of executing the function. Note that the inputs provided from the user appear in a different font and in blue.

```
>>> run_pizza_calculator()
Welcome to the COMP 202 fair pizza calculator!
Please chose one of the following modes:
A. "Quantity mode"
B. "Price mode"

Enter your choice: 1

This mode is not supported

>>> run_pizza_calculator()
Welcome to the COMP 202 fair pizza calculator!
Please chose one of the following modes:
A. "Quantity mode"
B. "Price mode"

Enter your choice: a

This mode is not supported
```

```
>>> run_pizza_calculator()
Welcome to the COMP 202 fair pizza calculator!
Please chose one of the following modes:
A. "Quantity mode"
B. "Price mode"

Enter your choice: A

You selected "Quantity mode"

Enter the diameter of the large pizza: 14
Enter the diameter of the small pizza: 8

To be fully satisfied you should order 4 small pizzas

>>> run_pizza_calculator()
Welcome to the COMP 202 fair pizza calculator!
Please chose one of the following modes:
A. "Quantity mode"
B. "Price mode"

Enter your choice: B

You selected "Price mode"

Enter the diameter of the large pizza: 12
Enter the price of the large pizza: 10.0
Enter the diameter of the small pizza: 6
Enter the number of small pizzas you'd like to buy: 2

The fair price to pay for 2 small pizzas is $5.0

>>> run_pizza_calculator()
Welcome to the COMP 202 fair pizza calculator!
Please chose one of the following modes:
A. "Quantity mode"
B. "Price mode"

Enter your choice: B

You selected "Price mode"

Enter the diameter of the large pizza: 14
Enter the price of the large pizza: 9.55
Enter the diameter of the small pizza: 8
Enter the number of small pizzas you'd like to buy: 4

The fair price to pay for 4 small pizzas is $12.47
```

For full credit you should never be repeating code, but rather calling helper functions! You are welcome to add additional functions if you think this can increase the readability of your code.

What To Submit

You must submit all your files on codePost (https://codepost.io/). The file you should submit are listed below. Any deviation from these requirements may lead to lost marks.

`vending_machine.py`
`pizza_calculator.py`
`README.txt` In this file, you can tell the TA about any issues you ran into doing this assignment. If you point out an error that you know occurs in your program, it may lead the TA to give you more partial credit.
This file is also where you should make note of anybody you talked to about the assignment. Remember this is an individual assignment, but you can talk to other students using the **Gilligan's Island Rule**: you can't take any notes/writing/code out of the discussion, and afterwards you must do something inane like watch television for at least 30 minutes.
If you didn't talk to anybody nor have anything you want to tell the TA, just say "nothing to report" in the file.